Constraint-Based Querying for Bayesian Network Exploration ⋆

Behrouz Babaki†, Tias Guns†, Siegfried Nijssen†,∗, and Luc De Raedt†

† KU Leuven, Celestijnenlaan 200A, 3000 Leuven - Belgium
∗ Universiteit Leiden, Niels Bohrweg 1, 2333 CA Leiden - The Netherlands

Abstract. Understanding the knowledge that resides in a Bayesian network can be hard, certainly when a large network is to be used for the first time, or when the network is complex or has just been updated. Tools to assist users in the analysis of Bayesian networks can help. In this paper, we introduce a novel general framework and tool for answering exploratory queries over Bayesian networks. The framework is inspired by queries from the constraint-based mining literature designed for the exploratory analysis of data. Adapted to Bayesian networks, these queries specify a set of constraints on explanations of interest, where an explanation is an assignment to a subset of variables in a network. Characteristic for the methodology is that it searches over different subsets of the explanations, corresponding to different marginalizations. A general purpose framework, based on principles of constraint programming, data mining and knowledge compilation, is used to answer all possible queries. This CP4BN framework employs a rich set of constraints and is able to emulate a range of existing queries from both the Bayesian network and the constraint-based data mining literature.

1 Introduction

Understanding a Bayesian network is not always easy. In particular users who are faced with a large network for the first time, or with networks that are dynamically updated when new data arrives, may not understand the knowledge encoded in such a network. It has been argued that BN's (especially those used for diagnosis) should be extensively evaluated before being used in practice [15].

While the Bayesian network literature already provides a set of queries and corresponding inference techniques that are helpful in gaining a better understanding of a network, most of the standard queries specify (and fix) the variables of interest, and then either ask for a most likely assignment to the variables or the computation of a particular probability.

This contrasts with common practice in the field of exploratory data mining, where one aims at understanding data by discovering and analyzing patterns. Since the seminal work on frequent itemset mining by Agrawal et al. [1], numerous techniques for exploratory mining of patterns under constraints have been

⋆ This work is published in proceedings of IDA 2015. The final publication is available at http://link.springer.com/chapter/10.1007/978-3-319-24465-5_2

developed [14]. The notions of frequency and pattern in constraint-based pattern mining actually correspond to the notions of probability and explanation in a Bayesian network. In pattern mining, one typically searches over a space of possible patterns. In Bayesian networks, this corresponds to searching over subsets of variables and their values. In this paper, we exploit the similarities between these two fields and introduce constraint-based queries for Bayesian networks.

The contribution of this paper is three-fold. First, inspired by constraint-based mining, we introduce an expressive set of exploratory queries for Bayesian networks. Secondly, we identify how these queries can be expressed as constraints over the variables and joint distribution of the Bayesian network. Finally, we show how these constraints can be expressed as a generic constraint program, combining ideas from constraint programming, itemset mining and knowledge compilation, in particular CP4IM [9] and arithmetic circuits (AC) [5]. Our method operates on the arithmetic circuit directly and can hence be applied to any graphical model that can be compiled into an AC. By doing so, we bridge the gap between constraint-based pattern mining and graphical models and contribute towards more intelligent analysis of Bayesian networks.

2 Examples of Bayesian Network Exploration

After introducing a Bayesian Network and BN pattern, we show examples of exploratory queries over a Bayesian network in an illustrative scenario.

Bayesian network pattern A *Bayesian network* $\mathcal{G}$ is a directed acyclic graph where each node represents a random variable X_i in $\mathcal{X} = \{X_1, \ldots, X_n\}$. Let $\mathrm{Pa}_{X_i}^{\mathcal{G}}$ denote the parents of X_i in $\mathcal{G}$. A joint distribution P over the set of variables $\mathcal{X}$ is said to factorize according to $\mathcal{G}$ if $P(X_1, \ldots, X_n)$ can be expressed as the product $\prod_{i=1}^{n} P(X_i|\mathrm{Pa}_{X_i}^{\mathcal{G}})$. We denote such a distribution by $P_{\mathcal{G}}$. We denote by $D(X_i)$ the *domain* of variable X_i, that is, the possible values the variable can take. An assignment of value x_i to variable X_i is denoted by $(X_i = x_i)$.

Definition 1 (BN pattern). *A pattern A over $P_{\mathcal{G}}$ is a partial assignment, that is, an assignment to a subset of the variables $\mathcal{X}$ in $\mathcal{G}$: $A = \{(X_1 = x_1), \ldots, (X_m = x_m)\}$, where the X_i are different variables and x_i is a possible value in $D(X_i)$.*

The probability of a pattern A, denoted by $P_{\mathcal{G}}(A)$, is $P((X_1 = x_1), \ldots, (X_m = x_m))$, that is, the marginal probability of the assignment. Our queries below will enumerate all satisfying BN patterns.

Example constraint-based queries Assume the manager of a New York car insurance company has just obtained a Bayesian network that describes the factors influencing cost claims of customers, cf. the in Figure 1. She wants to analyze the network to be able to assess costs, get more insight and provide recommendations to her personnel. In order to do so, she is interested in exploring patterns of interest in the network and poses a number of queries.

Q1. What are likely patterns given the evidence **PropertyCost = Million***?* These claims impose high costs on the company. Using a minimum probability of 0.015, she obtains 12 patterns, most of which contain either **SeniorTrain = False** or **Theft = False**. She is not interested in these and excludes them while lowering the threshold in the next query.

Q2. What are likely patterns that do not contain **SeniorTrain=False** *and* **Theft=False** *given the evidence* **PropertyCost = Million** *(with threshold* $\theta = 0.0105$*)?* She now gets 76 patterns, among which the pattern **A = {PropertyCost = Million, DrivingSkill = Substandard, DrivQuality = Poor, LiabilityCost = Thousand}**, which she finds interesting as it indicates a connection between high property cost, the driving capabilities of the customer, and the liability cost incurred. However, she wonders whether the pattern cannot be simplified.

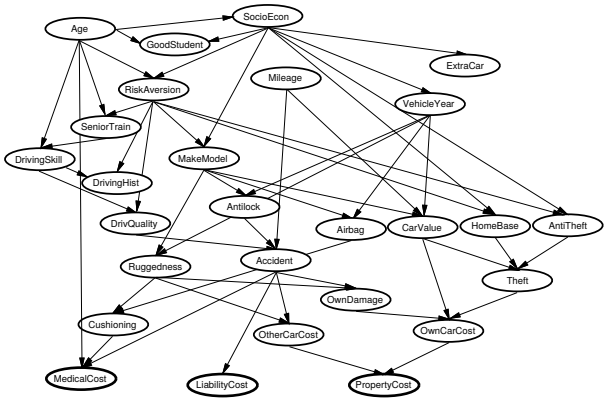

Fig. 1: The car insurance network [2].

Q3. Is there a simplification of pattern **A** *with the same probability?* She finds a variant of pattern **A** in which **DrivingSkill=Substandard** is removed, indicating that this assignment was implied and that there is some determinism in the network.

Now, turning her attention to the variable "Age", our manager wonders:

Q4. Are there any patterns that would allow to distinguish the age groups **Adolescent** *and* **Senior***?* She queries for patterns that have widely varying conditional probabilities when conditioned on each of these. After excluding the variables **SeniorTrain** and **GoodStudent**, which she already knows about, one of the top patterns is **{RiskAversion = Cautious, OtherCarCost = Thousand}**. Indeed the probability of having a cautious personality and incurring low third-party costs is nearly 6 times higher in senior customers.

Finally, a machine learning expert suggests to use a network trained on the company data instead (a simple naïve Bayes model). She wonders:

Q5. What are the patterns that have different probabilities according to the original and learned network? It turns out that the pattern **{Airbag=False, AntiLock=False, VehicleYear=Older}** has the largest difference of probabilities, hence the naïve Bayes model ignores the well-known relation between these three variables (namely older cars are rarely equipped with these safety components).

Q1: $probability(A, \mathcal{G}, \theta)$, $superset(A, \{\text{PropertyCost=Million}\})$
Q2: $probability(A, \mathcal{G}, \theta)$, $superset(A, \{\text{PropertyCost=Million}\})$, $exclude(A, \{\text{SeniorTrain, Theft}\})$
Q3: $maxprobability(A, \mathcal{G}, \theta')$, $free(A, \mathcal{G})$, $subset(A, \{\text{PropertyCost = Million, DrivingSkill = Substandard, DrivQuality = Poor, LiabilityCost =Thousand}\})$
Q4: $exclude(A, \{\text{SeniorTrain, GoodStudent}\})$, $ev\text{-}difference(A, \mathcal{G}, \{\text{Age=Adolescent}\}, \{\text{Age=Senior}\}, \beta)$
Q5: $difference(A, \mathcal{G}^1, \mathcal{G}^2, \beta)$

Table 1: The example queries expressed using constraints over pattern A.

3 BN query framework

We now formalize the queries above using constraints over patterns. Many other queries can be formulated this way, leading to a general querying framework.

Definition 2 (BN Pattern Query). *Consider a joint probability distribution $P_{\mathcal{G}}$ represented by a Bayesian network $\mathcal{G}$. We denote the set of all patterns of $\mathcal{P}_{\mathcal{G}}$ by $\mathcal{I}$. A BN pattern query $\mathcal{Q}$ is a tuple $(\mathcal{P}_{\mathcal{G}}, \mathcal{C})$ where $\mathcal{C} : \mathcal{I} \to \{0, 1\}$ is a conjunction of constraints over a pattern. Pattern A is a solution for $\mathcal{Q}$ if $\mathcal{C}(A) = 1$. The result of a query consists of all patterns that satisfy the constraints.*

The queries used in the examples in Section 2 are given in Table 1. Most constraints have close counterparts in the constraint-based pattern mining literature. The main difference is that the notion of *(relative) frequency* of a pattern in a database is replaced by the *probability* of the pattern in the BN. The constraints and their definitions are listed in Table 2 and explained below.

Probability constraint. Query Q1 requires that the probability of a pattern A according to $P_{\mathcal{G}}$ should be larger than a threshold θ. We call this constraint *probability*$(A, \mathcal{G}, \theta)$ and a pattern that respects it *θ-probable*. This definition is similar to the definition of a *frequency* constraint in frequent pattern mining.

Sub/superset and exclusion constraints Query Q1 also requires that patterns include given assignments. We enforce this with a superset constraint. Similarly, we can use *exclude*(A, V) to exclude variables from the pattern as in query Q2. The definition is given in Table 2, where vars(A) are the variables occurring in A.

Note that a superset constraint is conceptually similar to adding *evidence* in Bayesian networks, only that in our setting the computed probabilities will need to be normalized by the probability of the evidence to obtain the conditional probability.

Freeness, maximality and closedness constraint Query Q3 requires that a pattern does not contain redundant variable assignments. This is similar to the well-studied problem of simplifying explanations by excluding irrelevant variables [18], e.g. because of deterministic relations between assignments [7]. For pattern $A = B \cup C$ (where $B \cap C = \emptyset$), if variable assignments in B determine those in C, i.e., $P_{\mathcal{G}}(A) = P_{\mathcal{G}}(B)$, we consider those in the set C irrelevant. We

code	mathematical notation	CP formulation
PRB	$P_{\mathcal{G}}(A) \geq \theta$	$\mathsf{F}_1 \geq \theta$
MXP	$P_{\mathcal{G}}(A) \leq \theta$	$\mathsf{F}_1 \leq \theta$
SBS	$A \subseteq B$	$\forall i : \mathsf{Q}_i \neq 0 \implies (X_i = \mathsf{Q}_i) \in B$
SPS	$B \subseteq A$	$\forall (X_i = x_i) \in B : \mathsf{Q}_i = x_i$
EXC	$B \cap \text{vars}(A) = \emptyset$	$\forall X_i \in \text{vars}(B) : \mathsf{Q}_i = 0$
FRE	$\forall (X = x) \in A : P_{\mathcal{G}}(A \backslash (X = x)) > P_{\mathcal{G}}(A)$	$\forall i : \mathsf{Q}_i \neq 0 \rightarrow \left(\sum_j \mathsf{D}_{i,j}\right) > \mathsf{F}_1$
MAX	$\forall X \notin \text{vars}(A), \forall x \in D(X) : P_{\mathcal{G}}(A \cup \{(X = x)\}) < \theta$	$\forall i : \mathsf{Q}_i = 0 \rightarrow \bigwedge_j (\mathsf{D}_{i,j} < \theta)$
CLS	$\forall X \notin \text{vars}(A), \forall x \in D(X) : P_{\mathcal{G}}(A \cup \{(X = x)\}) < P_{\mathcal{G}}(A)$	$\forall i : \mathsf{Q}_i = 0 \rightarrow \bigwedge_j (\mathsf{D}_{i,j} < \mathsf{F}_1)$
DIF	$\lvert P_{\mathcal{G}^a}(A) - P_{\mathcal{G}^b}(A) \rvert \geq \beta$	$\lvert {\mathsf{F}_1}^a - {\mathsf{F}_1}^b \rvert \geq \beta$
DDF	$\lvert P_{\mathcal{G}}(A) - r_{\mathcal{D}}(A) \rvert \geq \beta$	$\lvert \mathsf{F}_1 - \mathsf{R} \rvert \geq \beta$
VDF	$\lvert P_{\mathcal{G}}(A \cup B)/P_{\mathcal{G}}(B) - P_{\mathcal{G}}(A \cup C)/P_{\mathcal{G}}(C) \rvert \geq \beta$	$\lvert {\mathsf{F}_1}^a/c_a - {\mathsf{F}_1}^b/c_b \rvert \geq \beta$

Table 2: constraints for BN pattern queries over patterns A, B, and C and network $\mathcal{G}$. Constraints are represented by three-letter codes PRB: *probability*$(A, \mathcal{G}, \theta)$; MXP: *maxprobability*$(A, \mathcal{G}, \theta)$; SBS: *subset*(A, B); SPS: *superset*(A, B); EXC: *exclude*(A, B); FRE: *free*$(A, \mathcal{G})$; MAX: *maximal*$(A, \mathcal{G}, \theta)$; CLS: *closed*$(A, \mathcal{G})$; DIF: *difference*$(A, \mathcal{G}^a, \mathcal{G}^b, \beta)$; DDF: *DB-difference*$(A, \mathcal{G}, \mathcal{D}, \beta)$; and VDF: *ev-difference*$(A, \mathcal{G}, B, C, \beta)$.

call a pattern *free* if none of its assignments is irrelevant. This definition is similar to the definition of free patterns in data mining [3]. In the presence of a *superset* constraint, the *free* constraint should only consider variables that are not required by the *superset* constraint.

Inspired by the related notions of maximality and closedness in frequent itemset mining, we introduce these for BN patterns too. They enforce that a pattern A does not have any superset that is θ-probable (i.e. *maximal*$(A, \mathcal{G}, \theta)$) or has the same probability as A (i.e. *closed*$(A, \mathcal{G})$).

Difference constraints Queries Q4 and Q5 both ask for patterns that demonstrate a difference between two probabilistic models. Let $P_{\mathcal{G}_1}(A)$ and $P_{\mathcal{G}_2}(A)$ be the probability of pattern A according to networks $\mathcal{G}_1$ and $\mathcal{G}_2$. The constraint *difference*$(A, \mathcal{G}_1, \mathcal{G}_2, \beta)$ requires that the difference of the probability of a pattern in these two networks is larger than β. In Q4, the two networks are obtained by assigning a variable in the original network to different values (B={Age=Adolescent} and C={Age=Senior} respectively). This can be formulated over network $\mathcal{G}$ using the constraint *ev-difference*$(A, \mathcal{G}, B, C, \beta)$. This constraints compares the conditional probability of A given evidence B or C.

Another variation can be used for testing the correlations between a Bayesian network and an actual dataset. This constraint compares the probability of a pattern in network $\mathcal{G}$ with the relative frequency of the corresponding itemset in the database $\mathcal{D}$. We call this constraint *DB-difference*$(A, \mathcal{G}, \mathcal{D}, \beta)$.

4 Formulating BN Pattern Queries As Constraint Programming Problems

In the Bayesian network literature, typically algorithms that search in the space of assignments are developed for specific constraints and scoring functions, which limits their general applicability (see Section 6 for a discussion of related work). In data mining, a recent trend is the use of generic solvers for handling a wide range of constraints in a uniform way.

We observe that there is a relationship between itemsets and BN patterns, as each variable assignment $(X_i = x_i)$ can be seen as one *item*, and hence a BN pattern can be seen as an itemset. Using this insight, we adapt the constraint programming for itemset mining framework [9] to reason over Bayesian networks. This framework has proven to support a wide range of constraints and exploratory queries over itemsets. Building on this framework, and hence the use of CP solvers, enable us to address a wide range of queries without the need to develop multiple specialized algorithms.

We first introduce the basics of constraint programming (CP), and explain how Bayesian networks can be encoded in CP in the form of an arithmetic circuit. We then explain how the constraints identified in Table 2 can be expressed in this framework.

Constraint Programming. Constraint programming is used to solve Constraint Satisfaction Problems (CSP) [17]. Constraint programming systems use generic solvers that search for solutions to a given CSP specification.

A CSP specification $\mathcal{P} = (\mathcal{V}, \mathcal{D}, C)$ consists of a set $\mathcal{V}$ of variables; $\mathcal{D}$ is the domain and maps every variable $V \in \mathcal{V}$ to a range of values $\mathcal{D}(V)$; and C is a set of constraints over subsets of $\mathcal{V}$. Generic solvers can be used to find all solutions that satisfy the constraints. These solvers use a combination of search (assigning a variable to a value) and propagation (per constraint, removing assignments from the domain that would violate that constraint) [17]. Many such generic yet efficient solvers exist.

BN pattern in Constraint Programming (CP) We can encode a BN pattern $A = \{(X_1 = x_1), \ldots, (X_m = x_m)\}$ in CP by introducing a CP variable Q_i for every network variable X_i. The domain of the CP variable Q_i consists of $|D(X_i)|+1$ values, where $D(X_i)$ is the set of possible values the BN variable X_i can take: value 0 to represent that X_i is not part of the pattern, e.g. it is marginalized over, and values $1 \ldots |D(X_i)|$ that each represent a possible assignment to the BN variable X_i.

BN pattern queries in CP Each of the constraints $\mathcal{C}$ of a BN pattern query $(\mathcal{P}_\mathcal{G}, \mathcal{C})$ can be formulated through CP constraints over the Q_i variables. We discuss this for each of the constraints in turn.

Probability constraint We will need to repeatedly compute the probability of a pattern, hence, we want this computation to be fast and ideally incremental. For this reason, we choose to first compile the BN into an Arithmetic Circuit (AC) [5]. Computing the probability of a partial assignment takes time polynomial to the size of the AC, though that size is exponential to the BN size in the worst case. Nevertheless, using ACs is generally recognized as one of the most effective techniques for exact computation of probabilities [5], especially when doing so repeatedly.

Figure 2 shows an example AC, consisting of product nodes, sum nodes, constants and *indicator variables* (ignore the square boxes for now). The Boolean indicator variables $\lambda_{i,j}$ indicate whether $(X_i = j)$. For ease of notation we will assume that the domain of the Bayesian variables is represented by consecutive integers starting from 1. To compute the probability of a partial assignment $\{(X_2 = 1), (X_3 = 2)\}$ we set $\lambda_{2,1} = 1, \lambda_{2,2} = 0, \lambda_{3,1} = 0, \lambda_{3,2} = 1$. X_1 is not in the pattern and needs to be marginalized away, so we set $\forall k \in D(X_1) : \lambda_{1,k} = 1$. Then, one computes the values of the internal AC nodes bottom-up, according to their operation ($\times$ or $+$). The value of the root node is the requested probability.

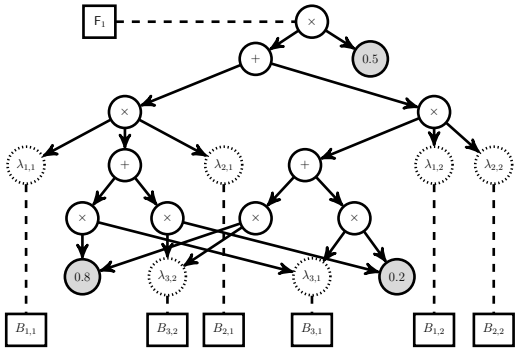

Fig. 2: Arithmetic circuit for a BN with 3 variables with domain $\{1, 2\}$ with X_1 the parent of X_2 and X_3. Square boxes represent CP variables.

This can be encoded in CP for arbitrary ACs: for each indicator variable $\lambda_{i,j}$ in the AC, we introduce a Boolean CP variable $\mathsf{B}_{i,j}$; the relation between the indicator variables and the CP variables Q_i is then modeled by the following constraints (recall that $\mathsf{Q}_i = 0$ means variable X_i is not in the pattern):

$$\mathsf{Q}_i = 0 \rightarrow \wedge_j \, (\mathsf{B}_{i,j} = 1) \qquad \forall i$$
$$\mathsf{Q}_i = k \rightarrow (\mathsf{B}_{i,k} = 1) \wedge (\wedge_{j \neq k} (\mathsf{B}_{i,j} = 0)) \qquad \forall i, \forall k \neq 0$$

We then introduce real-valued variable P, which will represent the computed probability. For this, we introduce an auxiliary real-valued variable F_v for each node in the circuit (round circles in Fig. 2). Assume each node has a unique identifier v, with the root node having identifier 1. Leaf nodes are either constants or indicator variables. The constants assign their corresponding F_v variable to a fixed value. For the indicator variables $\lambda_{i,j}$, the corresponding F_v variables are *channeled* to their Boolean counterparts $\mathsf{B}_{i,j}$ meaning they must take the same value (either 0 or 1). The internal nodes are then simply encoded by their operation, namely constraint $\mathsf{F}_v = \prod_{w \in \mathrm{Ch}(v)} \mathsf{F}_w$ for product nodes and constraint

$\mathsf{F}_v = \sum_{w \in \mathrm{Ch}(v)} \mathsf{F}_w$ for sum nodes, where $\mathrm{Ch}(v)$ are the identifiers of the children of node v in the AC.

Because of these constraints, when all Q_i (and hence B_i) variables are assigned, each F_v represents the value of that node of the AC, and the root node F_1 is the probability of the BN pattern. F_1 can then be used in a minimum probability constraint, see Table 2, right column.

Subset, superset and exclusion constraints Including evidence and excluding assignments in the pattern is done by constraining the relevant Q_i variables appropriately, as indicated in Table 2.

Freeness constraint To enforce this constraint, as explained in Section 3, we need to reason over the probability of subsets of a pattern. To do so, we use the observation that for an assignment $(X_i = k) \in A$: $P_{\mathcal{G}}(A \backslash \{(X_i = k)\}) = \sum_j P_{\mathcal{G}}((A \backslash \{(X_i = k)\}) \cup \{(X_i = j)\})$. Fortunately, using ACs we can efficiently compute these terms, as they correspond to *derivatives* of the function f encoded by the AC [5]. The latter work shows that for partial assignment A we have $P_{\mathcal{G}}((A \backslash \{(X_i = k)\}) \cup \{(X_i = j)\}) = \frac{\partial f}{\partial \lambda_{i,j}}(A)$. It was also shown that this can be computed for all nodes (and hence variables X) simultaneously using the derivatives of its parents in the AC, together with the values that we store in F_v variables.

To compute these derivatives, we introduce a real-valued CP variable D_v for every node v in the circuit. The value of D_v's corresponding to leaves $\lambda_{i,j}$, denoted by $\mathsf{D}_{i,j}$ for ease of notation, will represent the derivative of AC w.r.t $\lambda_{i,j}$: $\forall i, j\ \mathsf{D}_{i,j} = \frac{\partial f}{\partial \lambda_{i,j}}(A)$. Hence $\mathsf{D}_{i,j} = P_{\mathcal{G}}((A \backslash \{(X_i = k)\}) \cup \{(X_i = j)\})$.

Following the formulation in [5], the constraints below encode the computation of the D variables, where we denote by $\mathrm{Pa}^+(v)$ the identifiers of summation parents and by $\mathrm{Pa}^*(v)$ those of multiplication parents;

$$\mathsf{D}_v = \sum_{w \in \mathrm{Pa}^+(v)} \mathsf{D}_w + \sum_{w \in \mathrm{Pa}^*(v)} (\mathsf{D}_w \prod_{\substack{v' \in Ch(w) \\ v' \neq v}} \mathsf{F}_v) \qquad \forall v$$

$$\mathsf{D}_1 = 1$$

To formulate the *free* constraint from Table 2 over the CP variables, we use the fact that given $(X_i = k) \in A$: $P_{\mathcal{G}}(A \backslash (X_i = k)) = \sum_j \mathsf{D}_{i,j}$ and that $P_{\mathcal{G}}(A) = \mathsf{F}_1$.

Maximality and closedness constraints can be formulated using the same building blocks (c.f. table 2).

Difference constraints Comparing the probability of two networks over the same variables can be done by encoding the two ACs and formulating a mathematical constraint over the respective F_1 root node variables (Table 2).

Using CP allows us to easily mix different problems, such as combining the constraints of itemset mining in databases and BN's in a single CP model. The variable F_1 can be computed as before, while the relative frequency of a database

Network	#BN-n	c.Time(s)	#AC-n	#AC-e	θ	#Sols	s.Time(s)
HeparII	70	0.701	6963	13272	0.9	664	9.96
					0.8	24025	341.83
Win95pts	76	0.528	2786	6184	0.99	65	0.61
					0.95	214645	444.1
Insurance	27	0.374	34742	113788	0.9	12	2.76
					0.4	6662	383.66

Table 3: Probability queries over three benchmarks network, with **#BN-n**: number of BN nodes; **c.Time**: compilation time; **#AC-n/e**: number of AC nodes/edges; θ: probability threshold; **#Sols**: number of solutions, and **s.Time**: solving time.

over the same variables can be computed using a constraint programming for itemset mining formulation [9]. In Table 2 we materialize the relative frequency through a CP variable R.

As we have shown, many constraints over the pattern and the network can be readily formulated in CP. Furthermore, as these are standard CP constraints, existing CP solvers can be used to enumerate the satisfying BN patterns.

5 Experiments

We used the *ACE*[1] compiler (version 2) for generating arithmetic circuits from Bayesian networks. The networks were compiled with parameters " `-noTabular -cd06 -dtBnMinfill`". We used the *Gecode* [2] CP solver version 4.2.1. Experiments were run on Linux PCs with Intel 2.83GHz processors and 8GB of RAM.

Execution times for example queries To give an indication of execution times, we report the runtimes for the example queries of section 2 in Table 4a. The value of β for queries Q4 and Q5 was 0.08 and 0.25, respectively. The compilation time (not included in the reported runtimes) was 0.374 seconds.

To investigate the influence of size of BN and AC, we ran a simple query with only a $probability(A, \mathcal{G}, \theta)$ constraint on three benchmark networks[3]. Table 3 reports BN and AC size, θ threshold and runtimes. AC compilation time is small. Observe that in Table 3 the two larger networks have smaller AC's, because of their other structural properties (see [5] for more details). While bigger ACs require more runtime, the number of solutions has a major impact on runtime too. This can be controlled up to some extend by adding extra constraints.

Comparison with sampling An obvious alternative to our proposed method for executing itemset queries is to first sample a database from the joint distribution and then perform constraint-based itemset mining queries on the sampled

[1] http://reasoning.cs.ucla.edu/ace/
[2] http://www.gecode.org
[3] available at http://www.bnlearn.com/bnrepository/

Query	Q1	Q2	Q3
Time(s)	1.63	11.7	11.67
Query	Q4	Q5	
Time(s)	58.41	12.11	

(a)

#Samples	Precision	Recall	Time(s)
100	0.39	0.76	7.59
1000	0.73	0.94	20.08
10000	0.97	0.93	375.95

(b)

Table 4: (a) Execution times of example queries, and (b) Quality of results of sampling method as compared against the solutions of exact method.

database. Using this approach, one can execute the BN pattern queries using a constraint-based itemset mining system such as [9].

We investigate how this compares to our proposed method. We used two BNs: the first was the *insurance* network, which we will call BN1. The network BN2 is a naïve Bayes version of BN1 (With **PropertyCost** as root, and all non-cost observed variables as children) which we trained on 10000 samples from BN1. Compilation time for BN1 was 0.374 and 0.212 seconds for BN2. We then sampled a database of size 500 from BN2, which we call DB2.

In the approximate method, we sampled databases of varying sizes from BN1. We then searched for itemsets for which the relative frequency in the database and DB2 had a difference larger than 0.1. Table 4b presents the precision and recall of BN patterns found by the approximate method, compared to those found by our exact method. The results indicate that for a decent approximation, one needs to sample a large database which in turn leads to high computational costs. In comparison, the runtime of the exact method was 5.63 seconds.

6 Related work

Much attention in the Bayesian network literature has gone to the problem of finding explanations given some evidence. These *explanation queries* typically use a scoring function to find the best explanation. In contrast to queries like MAP and MPE, we do *not* fix which variables must be in or not in the pattern, instead we conceptually search over all possible marginalizations. There are other explanation queries that share this feature. These typically use specific scoring functions, such as the generalized Bayes factor of [19]. The explanation queries are constrained optimisation problems instead of enumeration problems. Our framework on the other hand is made for exploration queries and enumerates all satisfying BN patterns instead of computing the 'optimal' one.

There is also a body of work on discarding irrelevant variables from explanations [18, 6, 19, 11], as the *free* constraint does in our framework. In [6] each explanation found by a K-MPE algorithm is simplified by removing assignments that are considered irrelevant; [11] makes a trade-off between high probability and specificity. [18] proposes a definition for *relevance* and gives an algorithm that excludes irrelevant variables from the MAP assignments. This is a specific optimization query which is solved by a best-first-search algorithm.

Related to discriminating a BN network from a database, in [10] the authors search for subsets of variables rather than partial assignments. These *attribute*

sets are then used to modify the BN to better reflects the correlations present in the data. In other studies, a Bayesian network is used to *filter* itemsets or association rules found in a database. In [8], first an itemset mining algorithm is applied to a database to find a number of association rules, and then these rules are scored using the probability in the Bayesian and the concept of D-separation. In [12] the itemsets found by the well-known *apriori* algorithm are scored according to a Bayesian network, and the itemsets and attribute sets with highest scores are obtained in a post-processing step. The main difference with the discriminative setting considered in our work is that we compare patterns in the database and the network during search instead of post-processing them.

Our framework combines constraints with probabilistic computations. In similar spirit, there has been work on combining (deterministic) constraint networks with probabilistic networks [13]. The main difference is that in the resulting networks, all satisfying assignments are aggregated to compute a single probability value; on the other hand, we enumerate all possible *partial* assignments and compute their (marginal) probability.

7 Conclusions

We have investigated the problem of *exploring* Bayesian networks by querying for BN patterns (partial assignments) under constraints. The work is inspired by all the work on exploring data using constraint-based pattern mining techniques. We have shown that similar queries and constraints as used in the constraint-based pattern mining community can be used. This results in novel querying abilities for BNs. The proposed execution strategy is to compile the BN into an arithmetic circuit, and formulate and reason over that in a constraint programming framework. Such an approach supports a wide range of queries and constraints in a flexible and declarative manner.

Our work currently focusses on enumeration queries, as is typical in pattern mining. However, it could also be used in an optimisation setting over a scoring function, where its generality would allow one to add arbitrary constraints on top of the scoring function. In future work, the approach could also be adapted to problems beyond enumerating BN pattern queries, such as verifying monotonicity of Bayesian networks [16] or computing same-decision probability [4]. Our method may also be valuable for mining patterns over data, when evaluating the interestingness of the patterns using a BN (in our case, *during* search). Given the generality of the method, efficiency can be a concern though. Efficiency could be improved by using *global* constraints that can reason over the AC more efficiently, instead of using a decomposition over auxiliary variables F.

Acknowledgments. This work was supported by the European Commission under the project "Inductive Constraint Programming" contract number FP7-284715 and by the Research Foundation–Flanders by means of two Postdoc grants. We gratefully acknowledge useful comments and contributions from Guy Van den Broeck and Angelika Kimmig.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *SIGMOD Conference*, pages 207–216. ACM Press, 1993.
2. John Binder, Daphne Koller, Stuart J. Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2-3):213–244, 1997.
3. Jean-François Boulicaut and Baptiste Jeudy. Mining free itemsets under constraints. In *Proc. IDEAS '01*, pages 322–329. IEEE Computer Society, 2001.
4. Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. Algorithms and applications for the same-decision probability. *J. Artif. Intell. Res.*, 49:601–633, 2014.
5. Adnan Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, 2003.
6. Luis M. de Campos, José A. Gámez, and Serafín Moral. Simplifying explanations in bayesian belief networks. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(4):461–490, 2001.
7. Marek J Druzdzel and Henri J Suermondt. Relevance in probabilistic models:"backyards" in a "small world". In *Working notes of the AAAI–1994 Fall Symposium Series: Relevance*, pages 60–63, 1994.
8. Clément Fauré, Sylvie Delprat, Jean-François Boulicaut, and Alain Mille. Iterative bayesian network implementation by using annotated association rules. In *Managing Knowledge in a World of Networks, 15th International Conference, EKAW 2006, Proceedings*, volume 4248, pages 326–333, 2006.
9. Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining: A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983, 2011.
10. Szymon Jaroszewicz, Tobias Scheffer, and Dan A. Simovici. Scalable pattern mining with bayesian networks as background knowledge. *Data Min. Knowl. Discov.*, 18(1):56–100, 2009.
11. Johan Kwisthout. Most inforbable explanations: Finding explanations in bayesian networks that are both probable and informative. In *ECSQARU*, volume 7958, pages 328–339. Springer, 2013.
12. Rana Malhas and Zaher Al Aghbari. Interestingness filtering engine: Mining bayesian networks for interesting patterns. *Expert Syst. Appl.*, 36(3):5137–5145, 2009.
13. Robert Mateescu and Rina Dechter. Mixed deterministic and probabilistic networks. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):3–51, 2008.
14. Siegfried Nijssen and Albrecht Zimmermann. Constraint-based pattern mining. In Charu C. Aggarwal and Jiawei Han, editors, *Frequent Pattern Mining*, pages 147–163. Springer, 2014.
15. K Wojtek Przytula, Denver Dash, and Don Thompson. Evaluation of bayesian networks used for diagnostics. 60:1–12, 2003.
16. Merel T. Rietbergen, Linda C. van der Gaag, and Hans L. Bodlaender. Provisional propagation for verifying monotonicity of bayesian networks. In *ECAI*, volume 263, pages 759–764, 2014.
17. Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.
18. Solomon Eyal Shimony. The role of relevance in explanation I: irrelevance as statistical independence. *Int. J. Approx. Reasoning*, 8(4):281–324, 1993.
19. Changhe Yuan, Heejin Lim, and Tsai-Ching Lu. Most relevant explanation in bayesian networks. *J. Artif. Intell. Res. (JAIR)*, 42:309–352, 2011.